

Remarks

Reconsideration of the application is respectfully requested in view of the foregoing amendments and following remarks. Claims 1-8 and 10-23 are pending in this application. Claims 1, 8, 13 and 17 are independent. No claims have been allowed. Claims 1, 6, 7, 8, 13, 14, and 17 have been amended. Claim 20 has been canceled.

Request for Interview

If any issues remain in light of the following remarks, the Examiner is formally requested to contact the undersigned attorney prior to issuance of the next Office Action in order to arrange a telephonic interview. It is believed that a brief discussion of the merits of the present application may expedite prosecution. Applicant submits the following remarks so that the Examiner may fully evaluate Applicant's position, thereby enabling the interview to be more focused. **This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.**

Since Applicants are filing an RCE under 37 CFR 1.114 along with the enclosed response this request for interview is appropriate since "a request for an interview prior to the first Office action is ordinarily granted in continuing or substitute applications. See, MPEP § 713.02.

Applied Art

The Action relies on US Patent 6,389,464 to Krishnamurthy et al. ("*Krishnamurthy*") and U.S. Patent 6,546,419 to Humpleman et al. ("*Humpleman*").

Patentability of claims 1-8 and 10-23 over *Krishnamurthy* in view of *Humpleman* under 103(a) and over *Humpleman* alone under 102(e)

The Action rejects claims 1-8 and 10-23 under 35 USC § 103(a) as unpatentable over *Krishnamurthy* in view of *Humpleman* and under 35 USC § 102(e) over *Humpleman*. The Applicants respectfully traverse the rejection. The claims in their present form should be allowed over the applied art.

Section 103 Rejections

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. (MPEP § 2142.) Motivations to combine or modify references must come from the references themselves or be within the body of knowledge in the art. (See, MPEP § 2143.01.). However, the *Krishnamurthy* and *Humpleman*, both individually and in combination, fail to teach or suggest all the claim elements of the claims listed above.

Section 102 Rejections

For a 102(e) rejection to be proper, the applied art must show each and every element as set forth in a claim. (See, MPEP § 2131.01) However, *Humpleman* fails to do so.

Independent Claim 1

Amended claim 1 recites as follows:

A method of programmatically controlling a service of a logical device realized on a first computer from software programs running on a second computer, the method comprising:

from the first computer, obtaining at the second computer a service description message related to the service, the service description message detailing a service state table including at least one property associated with the service and a set of actions that can be invoked on the service via network data messages conveyed to the first computer via peer-to-peer networking connectivity over a data communications network connecting the first and the second computer, the set of actions comprising at least one action previously not available for invocation by the software programs running on the second computer;

based on the service description message, creating a service object corresponding to the service, the service object exposing a programming interface to access by software programs running on the second computer, the programming interface comprising an action-invoking member for invoking the set of actions listed in the service description including the at least one action previously not available for invocation by the software programs running on the second computer;

based on the service description message, converting a programmatic invocation of the action-invoking member of the programming interface by a software program running on the second

computer into a network data message for invoking one or more of the set of actions of the service via peer-to-peer networking connectivity over the data communications network; and
transmitting the network data message to the first computer to thereby invoke the one or more of the set of actions of the service.

Neither *Krishnamurthy* nor *Humpleman* teach or suggest every element of claim 1. For instance, both individually and in combination, they fail to teach or suggest, “*from the first computer, obtaining at the second computer a service description message related to the service, the service description message detailing a...a set of actions that can be invoked on the service..., the set of actions comprising at least one action previously not available for invocation by the software programs running on the second computer... based on the service description message, creating a service object corresponding to the service, the service object exposing a programming interface to access by software programs running on the second computer, the programming interface comprising an action-invoking member for invoking the set of actions listed in the service description including the at least one action previously not available for invocation by the software programs running on the second computer.*” The Action relies on *Humpleman*’s teaching of an interconnectivity model between two devices wherein the control of actions of one device by another is limited only to “overlapping” actions they may have in common. See, e.g., *Humpleman* at Col. 14, Lns. 6-19, which states as follows:

As discussed above in relation to FIG. 15, a first device B can access the INTERFACE.XML document of a second device A to examine the device capabilities and API interface details of the second device A and determine supported functionality and command details of the second device A. In particular, the first device B can determine overlapping, and therefore useable, methods supported by first device B and the second device A. FIG. 16 shows an example wherein a first server device B including an Application B accesses the INTERFACE-A.XML document of a second server device A including an Application A. **The first server device B includes a INTERFACE-B.XML document for comparison with that of a INTERFACE-A.XML document in the second server device A.** See, *Humpleman* at Col. 14, 2nd Para, Lns. 6-19.

Among other things, this fails to teach or suggest “*from the first computer, obtaining at the second computer a service description message related to the service... based on the service description message creating a service object corresponding to the service, the service object exposing a programming interface to access by software programs running on the second computer, the programming interface comprising an action-invoking member for invoking the set of actions listed in*

the service description including the at least one action previously not available for invocation by the software programs running on the second computer”, as claimed in Applicants’ claim 1. The Applicants’ claimed interconnectivity model is more comprehensive than what is taught by *Humpleman* and is able to allow control of one device by another including actions beyond those that are “overlapping”.

The applicants’ specification at Pg. 26, Lns. 5-12 with reference to FIG. 9 describes the action invocation as follows:

With reference to Figure 9, a preferred implementation 440 of the Rehydrator 410 is as an internal Microsoft Windows component that routes service control requests from the UPnP API to devices. Applications wishing to control a service on a UPnP device obtain a Service object through the UPnP API and use the methods of this object to query the state variables of the service and invoke its actions. Those methods use the private Rehydrator API to turn the service control requests into network messages that travel to the UPnP device. In this sense, the Rehydrator performs a mapping between API calls and network protocols.

Further at Pg. 26, Ln. 27 to Pg. 27, Ln. 2, of the Applicants’ specification, creation of the service-specific service object is described as follows:

When the Device Finder 450 creates a Device object, it invokes the Rehydrator 410 to create Service objects 460 for each of the service instances on that device. Each service instance supports a particular Service Control Protocol and the Rehydrator needs a description of this protocol in order to create a properly hydrated Service object.

The Service Control Protocol is declared in the Service Description. This document is passed to the Rehydrator as an IXMLDOMDocument interface pointer in the RehydratorCreateServiceObject() API call.

In addition to the creating the Service object, the Rehydrator sets up its internal data structures so that it can properly handle requests to control the service. Specifically, it creates a list of the properties and actions exported by the service and stores these as private data within the service object. The control and event subscription URLs as well as the service identifier and service type identifier are also stored in the service object.

Thus, in this manner, the claimed “service object” is created based on the “service description message” and as a result, software programs running on a second computer that acquire access to the “service object” can access all methods recited in the service description associated with a corresponding logical service running on a second computer “*including the at least one action previously not available for invocation by the software programs running on the second computer*” not just those methods that overlap with what was previously accessible to the software program on the

first computer.

However, what is taught in *Humpleman* is very different. Although, *Humpleman* teaches a first service accessing the “service description document” of a second service it is clear that *Humpleman* contemplates that this is “for comparison with that of a INTERFACE-A.XML document in the second server device A” in order to “determine overlapping, and therefore useable, methods supported by first device B and the second device A.” See, *Humpleman*, Col. 14, 2nd Para, Lns. 6-19. Thus, *Humpleman* fails to teach or suggest “... based on the service description message *creating a service object corresponding to the service*, the service object *exposing a programming interface to access by software programs running on the second computer*, the programming interface *comprising an action-invoking member for invoking the set of actions listed in the service description including the at least one action previously not available for invocation by the software programs running on the second computer*” as claimed in Applicants’ claim 1.

Furthermore, *Humpleman* teaches a Look-up Table (LUT) for “converting commands from a compiled C program code for Service B into XML form method requests.” See, *Humpleman*, Col. 13, Lns. 20-23. Thus, according to *Humpleman* the service description of a controlled device A may be obtained by a controlling device B to appropriately construct the LUT table to have the appropriately formatted commands sent to device A. No new API method calls are available to device B. Therefore, while *Humpleman* suggests a use for service description documents of a Service A by a Service B, (e.g., comparing INTERFACE-A.XML to INTERFACE-B.XML) it does not teach exposing a new API to Service B, based on INTERFACE-A.XML, for instance. Thus, no new API methods are available for service B as result of having access to the INTERFACE-A.XML, for instance. On the other hand, applicants’ claim 1 recites “based on the service description message *creating a service object corresponding to the service*, the service object *exposing a programming interface to access by software programs running on the second computer*, the programming interface *comprising an action-invoking member for invoking the set of actions listed in the service description including the at least one action previously not available for invocation by the software programs running on the second computer*”, which *Humpleman* fails to teach or suggest.

The Action further relies on the flow of events contemplated by *Humpleman* as stated in the following paragraphs:

A software agent, designated as Tool B, utilizes at least a subset of the objects and methods in the XCE definition for Service B and the CALL.DTD document, to generate a look-up table 62 for converting commands from a compiled C program code for Service B into XML form method requests. As such, the look-up table 62 provides conversion between a method invoked by Service B (e.g., "PLAY") and the XML document or message that carries the method call across the network interface to Service A, for example

Therefore, the API extensions provide for communication between various devices on the network using XML. In the example above, the program code 20 for Service B generates method calls to an API, and the API calls are converted to XML form to comply with the Web/Internet standard XML for inter-device communication. The XML method calls (messages) are sent to Service A over the network, and Service A reconverts the XML method calls from the network interface to program code API definitions for Service A. This conversion and re-conversion provides Web/Internet compatibility for diverse devices in the network with program code APIs which would otherwise require binary compatibility between different devices. Appendix 1 shows examples of the XML interface blocks utilizing the block diagrams in of FIG. 15. *See, Humpleman*, Col. 13, Lns. 31-45.

Upon examining the contents of the INTERFACE-A.XML document, the first server device B can create commands for sending to the second server device A in XML format as described above. Generally, the first server device B can interpret at least a portion of the contents of the INTERFACE-A.XML document that overlaps with a subset of the XCE definition used by the first and second server devices B and A as described above. See, Humpleman, Col. 14, Lns. 35-42.

However, "examining the contents of the INTERFACE-A.XML document, the first server device B can create commands for sending to the second server device A in XML format as described above" as taught by *Humpleman* is not the same as *"based on the service description message creating a service object corresponding to the service, the service object exposing a programming interface to access by software programs running on the second computer, the programming interface comprising an action-invoking member for invoking the set of actions listed in the service description including the at least one action previously not available for invocation by the software programs running on the second computer"* as claimed in Applicants' claim 1. This is so at least because, *Humpleman's* teaching of creating properly formatted "commands" does not lead one of ordinary skill in the art to *"creating a service object corresponding to the service, the service object exposing a programming interface to access by software programs running on the second computer."*

More particularly, according to *Humpleman*, a device description of a second device is used by a first device to compare it to its own device description for determining overlap methods and to properly format "commands" being directed to the second device. *See, e.g., Humpleman*, Col. 13, Lns.

3-6 stating: (“The INTERFACE-A.XML can also be used by a foreign Application such as Service B to determine the message format for Service A before communicating with Service A). However, such “commands” do not include “at least one action previously not available for invocation by the software programs running on the second computer.”

Therefore, what is taught by *Humpleman* does not lead one of ordinary skill in the art to “*based on the service description message creating a service object corresponding to the service, the service object exposing a programming interface to access by software programs running on the second computer, the programming interface comprising an action-invoking member for invoking the set of actions listed in the service description including the at least one action previously not available for invocation by the software programs running on the second computer”*. *Krishnamurthy* too fails to teach or suggest the same.

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 1. Thus, claim 1 in its present form should be allowed over the cited art.

Independent Claim 17:

Claim 17 was amended to include elements previously recited in dependent claim 20. The amended claim 17 recites as follows:

A software module carried on a tangible computer-executable software carrying medium, the software module operable for:

providing programmatic control of a logical device running on a computing device by an application software on a different computing device over a data communications network by exposing a programming interface for providing programmatic logical device service control via peer networking connectivity, the programming interface comprising:

an invoke action method member having parameters for passing an action identifier, action arguments and action return value;

wherein an implementation of the invoke action method member in the software module converts an invocation of the invoke action method member into an exchange of text messages with the logical device via peer networking connectivity based on a service description obtained from the logical device to control a service of the logical device; and

a logical device state call back method member having parameters for passing a reference to a call back interface for reporting change of the logical device's state to other registered devices on the network.

Neither *Krishnamurthy* nor *Humpleman* teach or suggest every element of claim 17. For instance, both individually and in combination, they fail to teach or suggest “programming interface comprising: an invoke action method member having parameters for passing an action identifier, action arguments and action return value ...a logical device state call back method member having parameters for passing a reference to a call back interface for reporting change of the logical device's state to other registered devices on the network.” Applicants' claim 17 recites one “programming interface comprising: an invoke action method member” and “a logical device state call back method member... passing a reference to a call back interface for reporting change of the logical device's state to other registered devices on the network.” The applicants' specification at Pg. 131, Lns. 10-25 describes such an exemplary “call back interface” as follows:

For each service in a device, a description message contains an eventing URL (eventSubURL sub element of service element in the device description) and the UPnP service identifier (serviceId sub element in service element in device description). To subscribe to eventing for a particular service, a subscription message is sent to that service's eventing URL. The message contains that service's identifier as well as a delivery URL for event messages. A subscription message may also include a requested subscription duration.

To subscribe to eventing for a service, a control point sends a request with method SUBSCRIBE and NT and CALLBACK headers in the following format. Values in *italics* are placeholders for actual values.

SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
CALLBACK: <delivery URL>

The Action relies on *Humpleman* and *Krishnamurthy* and notes that Applicants' “claims 17-23 list all the same elements of claim 1-7, but in software module form rather than method form.” *See*, Action, Pg. 6. Applicants respectfully disagree. Claim 17 has now been amended to add elements of previous listed claim 20 (now canceled) to bring to the Examiner's attention that the element of applicants' claim 17 which recites “*a logical device state call back method member having parameters for passing a reference to a call back interface for reporting change of the logical device's state to other registered devices on the network*” is not found anywhere in Applicants' claims 1-7 and as such cannot be rejected by merely asserting that they are the same as what in claims 1-7.

Nevertheless, neither *Krishnamurthy* nor *Humpleman* teach or suggest “*programming interface comprising: an invoke action method member having parameters for passing an action identifier, action arguments and action return value*” and “*a logical device state call back method member having parameters for passing a reference to a call back interface for reporting change of the logical device’s state to other registered devices on the network.*” Applicants’ claim recites a “programming interface” having method members for not only invoking methods of target device service, but, also, for requesting and receiving notices of events related to changes in state of the target service by having “*a call back interface for reporting change of the logical device’s state to other registered devices on the network.*” In this manner, a single point of control for affecting change on a service and for receiving notices of events related to changes in a service’s state allows a control point to be aware of changes brought about in the service by itself and other control points. Moreover, this gives control of how and whether such notifications are to be made “*a reference to a call back interface for reporting change.*” No such comprehensive control is taught or suggested by *Krishnamurthy* or *Humpleman*, both individually or in combination.

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually or when combined, fail to describe at least one feature recited in claim 17. Thus, claim 17 in its present form should be allowed over the cited art.

Independent Claim 8:

Claim 8 recites as follows:

In a networking environment providing peer-to-peer connectivity between logical devices on separate computing machines on a data communications network in accordance with a control protocol, a user-operated control device comprising:

a rehydrating module operable for:

receiving the control protocol defining an exchange between a control point and a controlled logical device service in which the controlled logical device service furnishes the control protocol to the control point in a service description message, the service description message specifying a set of actions invocable on the controlled logical device service, the set of actions comprising at least one action previously not available for invocation by the control point; and

the rehydrating module further operable for creating a service object for exposing an application programming interface based on the service description message for invoking set of

actions specified in the service description message including the at least one action previously not available for invocation by the control point;

the application programming interface exposed by the rehydrating module to access from application software running on the user-operated control device, the application programming interface having an invoke action member operable for invoking the set of actions specified in the service description message including the at least one action previously not available for invocation by the control point;

invoke action member-implementing code of the rehydrating module operating responsive to an invocation of the invoke action member to generate a peer networking data message to cause the controlled logical device service to perform a respective action of the controlled logical device service; and

converting code of the rehydrating module operating to construct the peer networking data message based on the control protocol obtained via the service description message.

Neither *Krishnamurthy* nor *Humpleman* teach or suggest every element of claim 8. For instance, both individually and in combination, they fail to teach or suggest, “A rehydrating module operable for ...creating a service object for exposing an application programming interface based on the service description message for invoking set of actions specified in the service description message including the at least one action previously not available for invocation by the control point.”

The Action relies on *Humpleman*'s teaching of an interconnectivity model between two devices wherein the control of actions of one device by another is limited only to “overlapping” actions they may have in common. See, e.g., *Humpleman* at Col. 14, Lns. 6-19. For the reasons noted above with respect to claim 1, *Humpleman* fails to teach or suggest “a rehydrating module operable for ...creating a service object for exposing an application programming interface based on the service description message for invoking set of actions specified in the service description message including the at least one action previously not available for invocation by the control point.” *Krishnamurthy* too fails to teach or suggest the same.

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 8. Thus, claim 8 in its present form should be allowed over the cited art.

Independent Claim 13:

Claim 13 recites as follows:

A computer-readable data carrying medium having software program code carried thereon, the software program code comprising:

a programmatic peer networking device service control module providing programmatic control of logical device services on a computing device by application software running on a different computing device on a data communications network via a peer-to-peer networking connectivity service control protocol;

an application programming interface for access by the application software, the application programming interface exposed by a service object created based on a service description message comprising a set of actions invocable on the logical device services, the service object created by the programmatic peer networking device service control, the application programming interface being a run-time dispatch interface having an invoke service action method member, the invoke service action method member accepting an action identifier, ingoing action arguments, outgoing action arguments, and action return value as parameters upon invocation by the application software, wherein the action identifiers, the ingoing action arguments, the outgoing action arguments and the action return values correspond to those of the set of actions specified in the service description message and are specified therein, the set of actions including at least one action not available for invocation by application software previous to the creation of the service object; and

invoke service action method member-implementing code of the programmatic peer networking device service control module operating responsive to an invocation of the invoke service action method member on the application programming interface by the application software to exchange data messages with a logical device service of a separate computing device on the data communications network in accordance with the peer-to-peer networking connectivity service control protocol so as to invoke an action of the logical device service including the at least one action not available for invocation by application software previous to the creation of the service object as per the parameters of the invoke service action method member and pass outgoing action arguments and action return value from the logical device service back to the application software.

Neither *Krishnamurthy* nor *Humpleman* teach or suggest every element of claim 13. For instance, both individually and in combination, they fail to teach or suggest “an application programming interface for access by the application software, the application programming interface exposed by a service object created based on a service description message...to invoke an action of the logical device service including the at least one action not available for invocation by application software previous to the creation of the service object.”

The Action relies on *Humpleman*’s teaching of an interconnectivity model between two devices wherein the control of actions of one device by another is limited only to “overlapping” actions they may have in common. See, e.g., *Humpleman* at Col. 14, Lns. 6-19. At least for the reasons presented above with respect to claims 1 and 8, this does not teach or suggest “an application programming interface for access by the application software, the application programming interface exposed by a service object created based on a service description message...to invoke an action of the logical

device service including the at least one action not available for invocation by application software previous to the creation of the service object. ”

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 13. Thus, claim 13 in its present form should be allowed over the cited art.

Dependent Claim 6:

Claim 6 recites as follows:

The method of claim 1 wherein the programming interface further has a service state-querying method member, the method further comprising:
responsive to programmatic invocation of the service state-querying method member by the software programs running on the second computer, obtaining state data of the service via peer-to-peer networking connectivity over the data communications network
updating the service state table; and
returning the state data to the invoking software program.

Claim 6 depends from claim 1 and as such, is allowable over *Krishnamurthy* and *Humpleman* for the reasons listed above with reference to claim 1. However, claim 6 sets forth independently patentable combinations of elements. For instance, claim 6 recites “programming interface further has a service state-querying member” wherein “responsive to programmatic invocation of the service state-querying member by the software programs running on the second computer, obtaining state data of the service via peer-to-peer networking connectivity over the data communications network updating the service state table; and returning the state data to the invoking software program.” The Action relies on *Humpleman*, Col. 5, Lns. 55-67 and Col. 6, Lns. 1-60. *See*, Action Pg. 5. However, what *Humpleman* teaches is very different. For instance, *Humpleman* at Col. 5, Lns. 55-67 and Col. 6, Lns. 1-7 states as follows:

Referring still to FIG. 4, a server device 14 may include one or more server control programs 20 to control the server hardware for providing a service. The GUI interface 18 from the GCO 22 of the server device 14 provides interface to the server device control programs 20. The server device 14 may also include control state data 26 indicating the control status of the server device 14 and server device hardware in providing a requested service.

For example, the control state data 26 can include the status of control information in the GUI 18 for the server device 14, such as timer setup for a recording action in a VCR server

device. The control state data 26 is stored in the controlled server device 14, and displayed to a user through the GUI 18 of the server device 14 at the controlling client device 12, for user control of the server device 14. Preferably, the controlling client device 12 for displaying the GUI 18 of the server device 14 does not retain knowledge of the control state data 26 for the controlled server device 14.

However, Humpleman's "control state data" is presented via a Graphical User Interface (GUI) from a controlled "server device" to a client. Thus, Humpleman fails to teach or suggest "a programming interface" wherein "wherein the programming interface further has a service state-querying method member." The GUI of Humpleman (e.g., at 18 in FIG. 4 of Humpleman) does not teach or suggest "a programming interface... wherein the programming interface further has a service state-querying method member" as claimed in applicants' claim 1. More particularly, a GUI displayed at a client with "control state data" does not allow for the request-response operability provided by "a programming interface... wherein the programming interface further has a service state-querying method member." Humpleman, recites its API for control programs at Col. 21, Ln. 47 to Col. 24, Ln. 37 in great detail, but fails to mention "a service state-querying method member", let alone a method whereby "responsive to programmatic invocation of the service state-querying method member ... updating the service state table; and returning the state data to the invoking software program." *Krishnamurthy* too fails to teach the same.

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 6. Thus, claim 6 in its present form should be allowed over the cited art.

Dependent Claim 7:

Claim 7 recites as follows:

The method of claim 1 wherein the programming interface further has a service state-querying method member that accepts an invocation parameter indicative of a state data variable of the service, the method further comprising:

responsive to programmatic invocation of the service state-querying method member by the software programs running on the second computer, obtaining a value of the state data variable of the service via peer-to-peer networking connectivity over the data communications network

updating the service state table with the obtained value; and

returning a datum indicative of the value of the state data variable to the invoking software program.

Claim 7 depends from claim 1 and, as such, is allowable over *Krishnamurthy* and *Humpleman* for the reasons listed above with reference to claim 1. However, claim 7 sets forth independently patentable combinations of elements. For instance, claim 7 now recites “the programming interface further has a service state-querying method member that accepts an invocation parameter indicative of a state data variable of the service...responsive to programmatic invocation of the service state-querying method member by the software programs running on the second computer, obtaining a value of the state data variable of the service via peer-to-peer networking connectivity over the data communications network.” The Action again relies on *Humpleman*, Col. 5, Lns. 55-67 and Col. 6, Lns. 1-60. *See*, Action Pg. 5. However, as noted above, *Humpleman*’s GUI fails to teach or suggest a “programming interface...wherein the programming interface further has a service state-querying method member that accepts an invocation parameter indicative of a state data variable of the service...responsive to programmatic invocation of the service state-querying method member by the software programs running on the second computer, obtaining a value of the state data variable of the service via peer-to-peer networking connectivity over the data communications network.” *Krishnamurthy* too fails to teach the same.

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 7. Thus, claim 7 in its present form should be allowed over the cited art.

Dependent Claim 18:

Claim 18 recites as follows:

The software module of claim 17 wherein the programming interface further comprises a state variable querying method member having parameters for passing a state variable identifier and state variable value relating to a logical device state variable.

Claim 18 depends from claim 17 and, as such, is allowable over *Krishnamurthy* and *Humpleman* for the reasons listed above with reference to claim 17. However, claim 18 sets forth independently patentable combinations of elements. For instance, claim 18 now recites “the

programming interface further comprises a state variable querying method member having parameters for passing a state variable identifier and state variable value relating to a logical device state variable.” At least for the reasons listed above with respect to claims 6 and 7 *Humpleman* and *Krishnamurthy* fail to teach or suggest “a state variable querying method member having parameters for passing a state variable identifier and state variable value relating to a logical device state variable.”

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 18. Thus, claim 18 in its present form should be allowed over the cited art.

Dependent Claim 21:

Claim 21 recites as follows:

The software module of claim 17 wherein the programming interface further comprises status method members having parameters for returning a value indicative of a status of controlling the service of the logical device.

Claim 21 depends from claim 17 and, as such, is allowable over *Krishnamurthy* and *Humpleman* for the reasons listed above with reference to claim 17. However, claim 18 sets forth independently patentable combinations of elements. For instance, claim 21 now recites “the programming interface further comprises status method members having parameters for returning a value indicative of a status of controlling the service of the logical device.” At least for the reasons listed above with respect to claims 6, 7, and 18, *Humpleman* and *Krishnamurthy* fail to teach or suggest “status method members having parameters for returning a value indicative of a status of controlling the service of the logical device.”

Thus, at least for the reasons listed above, the cited references, *Krishnamurthy* and *Humpleman*, both individually and when combined, fail to describe at least one feature recited in claim 21. Thus, claim 21 in its present form should be allowed over the cited art.

Dependent claims 2-5, 10-12, 14-16, 19 and 22-23:

Claims 2-5 ultimately depend on claim 1. Thus, at least for the reasons set forth above with respect to claim 1, claims 2-5 should also be in condition for allowance. Claims 10-12 ultimately

depend on claim 8. Thus, at least for the reasons set forth above with respect to claim 8, claims 10-12 should also be in condition for allowance. Claims 14-16 ultimately depend on claim 13. Thus, at least for the reasons set forth above with respect to claim 13, claims 14-16 should also be in condition for allowance. Claims 19 and 22-23 ultimately depend on claim 17. Thus, at least for the reasons set forth above with respect to claim 17, claims 19 and 22-23 should also be in condition for allowance. Each of these claims also set forth independently patentable combinations of elements.

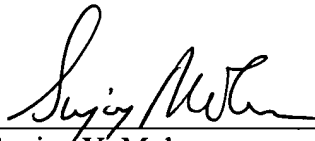
Conclusion

The claims should be allowed. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By



Sunjay Y. Mohan
Registration No. 56,739

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 228-9446